Attorney Docket Number: FSP0050
Title: ERROR HANDLING SCHEME FOR TIME-CRITICAL PROCESSING ENVIRONMENTS
Application Number: 10/827,158

REMARKS

In an office action mailed on 12/04/2006, claims 1-11 are rejected under 35 USC 102(e) as anticipated by Wong, US 6871350.

One aspect of independent claim(s) 1, 5, and 9 is that as a result of detecting a device error, the jump table is reconfigured to return without invoking the command processing logic and in a manner that indicates that the commands were successfully carried out.

There is nothing in Wong that teaches or suggests that the jump table is reconfigured as a result of detecting a device error. Wong does not teach, suggest, or imply that the caller, device driver, or anything else would reconfigure the device driver jump table when a device error is detected.

Wong suggests nothing along the lines that as a result of a device error, the jump table would return without first invoking the device driver code, and further, that the jump table would return in a manner that indicates that invoked device driver commands were invoked successfully (when in fact, they were not). None of these features, all recited in the claims, are taught, suggested, or even implied by Wong.

The Office Action cites Wong, col. 5, lines 57-62 and Fig. 4, which teaches that no kernel resources can be held when GRE 64 calls the user-mode graphics driver 62 (to ensure that a driver error cannot cause the system to wait indefinitely). In this embodiment, the KM Proxy Layer 70 releases and reacquires resources as necessary. Here Wong merely teaches that the described architecture helps prevent system hangs if the driver crashes. There is nothing here, explicit or implied, about the jump table being reconfigured as a result of detecting a device error, or about the jump table returning without first invoking the device driver code, or that the jump table would return in a manner that indicates that invoked device driver commands were invoked successfully.

One aspect of the dependent claims is continuing to provide commands to configure a display from to the driver logic after detection of the error. The Office Action cites Wong, col. 6, lines 27-34, which teaches that when a user-mode graphics driver 62 calls back via an EngXX function, the call is serviced by a user-mode EngXX stub in the UM Proxy Layer 74, which in turn calls kernel-mode via the corresponding call (depicted as NtGdiEngXX) in FIG. 2) provided by the KM Proxy Layer 70 (104). Here, Wong merely describes the callback sequence for the user mode device driver, where the call is stubbed to kernel mode code. Wong says and implies nothing about continuing to invoke the driver logic with display frame configuration commands, even after detection of a device error. The citation does not even appear to be remotely relevant to this feature of the claims.

Another aspect of the dependent claims is that the provider of the commands acts to correct the device error. The Office Action cites Wong, col. 6, lines 35-46, which teaches that the KM Proxy Layer 70 invokes the Proxy Layer Object Manager 76 to find and substitute the original kernel-mode GRE 64 object (108-112). In addition, for any user-mode memory buffers that are passed (114), the Proxy Layer Object Manager 76 either secures it or makes a kernel-mode copy (116). Finally, it performs parameter checking such as checking for illegal, out-of-range, or inconsistent values (120-122). After the Proxy Layer Object Manager 76 has finished these tasks, the KM Proxy Layer 70 calls the kernel-mode EngXX function in GRE 64 (128). If there are any errors, they are returned (110, 118, 124, 126). Here Wong merely describes moving data objects and buffers from user mode to kernel mode, and checking the parameters of the commands themselves for proper value. There is nothing about the provider of the commands acting to correct a device error. Furthermore, notice that Wong is actually teaching away from the present claims by teaching, contrary to the claims, that the errors are returned to the caller, instead of an indication of successful invocation.

Another aspect of the dependent claims is that as a result of the correcting of the error, the jump table is reconfigured to again direct commands to the command

processing logic. As previously noted, there is nothing anywhere in Wong about reconfiguring the jump table in light of a device error. Nor, it follows, is there anything in Wong about reconfiguring the table again to enable driver processing once the error is corrected.

In view of the above amendments and remarks, applicant believes that this application is now in condition for allowance. Applicant respectfully requests that a Notice of Allowability be issued covering the pending claims. If the Examiner believes that a telephone interview would in any way advance prosecution of the present application, please contact the undersigned.

Signature /Charles A. Mirho/ Date: 2/22/2007
Charles A. Mirho
Reg. 41,199
Attorney for Applicant

Address all correspondence to:
FSP LLC
Attn: Charles A Mirho
P.O. Box 890
Vancouver, WA 98666-0890
USA

Phone: 360-737-1748
Fax: 360-294-6426